arXiv:2508.04946v1 [cs.LG] 7 Aug 2025

# REINA: Regularized Entropy Information-Based Loss for Efficient Simultaneous Speech Translation

**Nameer Hirschkind***, **Joseph Liu***, **Xiao Yu, Mahesh Kumar Nandwana**

Roblox

{nhirschkind, josephliu, xyu, mnandwana}@roblox.com

**Abstract**

Simultaneous Speech Translation (SimulST) systems stream in audio while simultaneously emitting translated text or speech. Such systems face the significant challenge of balancing translation quality and latency. We introduce a strategy to optimize this tradeoff: wait for more input only if you gain information by doing so. Based on this strategy, we present Regularized Entropy INformation Adaptation (REINA), a novel loss to train an adaptive policy using an existing non-streaming translation model. We derive REINA from information theory principles and show that REINA helps push the reported Pareto frontier of the latency/quality tradeoff over prior works. Utilizing REINA, we train a SimulST model on French, Spanish and German, both from and into English. Training on only open source or synthetically generated data, we achieve state-of-the-art (SOTA) streaming results for models of comparable size. We also introduce a metric for streaming efficiency, quantitatively showing REINA improves the latency/quality trade-off by as much as 21% compared to prior approaches, normalized against non-streaming baseline BLEU scores.

## 1 Introduction

Simultaneous Speech Translation (SimulST) involves real-time translation of speech in one language into text in another. This extends the simpler speech-to-text-translation (S2TT) task, which involves translation with the full context of an entire speech clip. While S2TT allows for offline applications, conversational environments such as voice or video chat necessitate SimulST models to facilitate real-time communication across language barriers.

Recently, End-to-end (E2E) S2TT models have largely superseded traditional cascaded approaches, which link separate Automatic Speech Recognition (ASR) and Machine Translation (MT) systems. E2E models mitigate error propagation and reduce latency by directly mapping source speech to target text (Communication et al. 2023; Peng et al. 2024; Radford et al. 2023; Puvvada et al. 2024).

Rather than training SimulST models from scratch, most works take advantage of advances in S2TT research by adapting a non-streaming S2TT model into a SimulST model (Communication et al. 2023; Chen et al. 2024; Zhang et al. 2024; Papi et al. 2024). To transition from non-streaming S2TT to SimulST introduces the challenge of balancing translation quality and latency. This requires a policy to decide whether to wait for more input (READ) or generate output (WRITE) (Gu et al. 2017). This problem is particularly difficult in the translation setting because different languages can have different word orderings, requiring differing amounts of context before a suitable translation can occur. Many approaches have been formulated to determine this READ/WRITE policy, from baking the policy into the model architecture itself via monotonic attention mechanisms (Arivazhagan et al. 2019; Ma et al. 2020b; Communication et al. 2023), or having a separate module to dictate the policy (Chen et al. 2024). However, these existing approaches suffer from issues including poor translation quality compared to non-streaming models and expensive, numerically unstable training (Communication et al. 2023; Chen et al. 2024).

In this paper, we address the problem of efficiently training high quality SimulST models. The major contributions of this paper can be summarized as follows:

**New Policy Training Technique.** We propose Regularized Entropy INformation Adaptation (REINA), a new technique for policy training that can efficiently convert non-streaming Speech-to-Text Translation (S2TT) models into simultaneous S2TT (SimulST) models. REINA is guided by an approximation of mutual information derived from the S2TT model's log probabilities on partial versus full audio and is shown to produce higher quality policies than existing methods.

**Trained with open-source data.** We train an E2E S2TT model with REINA on 130k hours of open-source data. Based on empirical studies, this model achieves SOTA streaming translation performance.

**Streaming efficiency evaluation metric.** We propose a new evaluation metric to better compare SimulST models. This metric normalizes the streaming translation quality against the performance of the underlying non-streaming model, allowing for a fairer assessment of the capabilities of the streaming policy itself.

## 2 Related Work

While learning adaptive policies for SimulST is a fairly new research area, it builds on a rich body of non-streaming

*These authors contributed equally.

S2TT work. In this section, we outline the S2TT foundation for SimulST and then move into a discussion of SimulST policies.

**Training Speech to Text Translation Models** The literature around S2TT contains many large-scale, powerful models including Whisper (Radford et al. 2023), SeamlessM4T (Communication et al. 2023), Canary (Puvvada et al. 2024), and the Open Whisper-style Speech Model (OWSM) (Peng et al. 2024). These models vary in architecture (e.g., Whisper's Transformer, OWSMv3.1's E-Branchformer) and training data scale, ranging from Canary's 86k hours (leveraging pseudo-labels) and OWSM's 180k hours of public data to Whisper's 680k hours of web data and SeamlessM4T's 600k hours of synthetically aligned data.

Due to the relative scarcity of parallel ST data compared to ASR or MT corpora, multi-task learning (MTL) is widely adopted (Ye, Wang, and Li 2022; Chen et al. 2024; Communication et al. 2023). Auxiliary tasks like ASR and neural machine translation (NMT) are jointly trained with S2TT to improve representations and leverage abundant text or speech data. Furthermore, contrastive learning techniques, such as in ConST (Ye, Wang, and Li 2022), are used to explicitly bridge the modality gap between speech and text representations by encouraging similarity between corresponding speech segments and their transcriptions. There is a notable gap in the literature between industry work leveraging massive proprietary datasets and less resourced research making heavy use of MTL to get the most out of smaller data scales. As OWSM (Peng et al. 2024) bridges this gap for the non-streaming setting, we aim to do the same for SimulST. We are one of the first SimulST works to leverage large-scale open source data that we train on with an MTL framework including MT and ASR tasks.

**Streaming Policy Learning** Transitioning S2TT models to SimulST introduces the challenge of learning a READ/WRITE policy that balances translation quality and latency. Fixed policies like wait-k are simple to implement but are usually suboptimal due to the mismatch between the sampling rate of the input audio frames and the frequency of outputted words (Ma, Pino, and Koehn 2020). That being said, recent works like SimulS2S-LLM (Deng et al. 2025a) have used wait-k with some success. Adaptive polices based on heuristics such as attention matrix weights as seen in EdAtt (Papi, Negri, and Turchi 2023) also been proposed, which are usually better.

On the other hand, adaptive learnable policies dynamically adjust decisions based on context. Prior works have integrated the policy within the model architecture, such as Transducer models (Graves 2012; Xue et al. 2022), which inherently support streaming via monotonic alignment, or models using monotonic attention mechanisms like MMA (Arivazhagan et al. 2019) or EMMA (Communication et al. 2023). Monotonic alignment methods afford greater expressivity, but they tend to be excessively expensive to compute at train time and suffer from both poor numerical stability and difficulty in converging. Our preliminary investigations validate this claim.

Other SimulST works avoid complex, explicit policies, instead generating aligned data with which to directly train SimulST models (Labiausse et al. 2025; Fu et al. 2025; Deng et al. 2025b). These works often use existing models such as NMT models (Labiausse et al. 2025) or LLM's (Fu et al. 2025; Deng et al. 2025b) as teachers to create synthetically aligned data for streaming training. Such models can afford simpler architectures without policy networks. That said, SimulST models deriving their policies from generated data are often limited in their streaming performance based on the quality of the teacher model.

**Explicit policy training** Other adaptive strategies decouple the policy from the translation model (Chen et al. 2024; Gu et al. 2017; Zhang et al. 2024). Some leverage signals from pre-trained offline models, such as using reinforcement learning (RL) to directly optimize the quality-latency tradeoff (Gu et al. 2017). Although these methods simplify the learning problem by decoupling the policy from the translation model, they require explicit supervision from a suitable metric and are often suboptimal. RL is hard to stabilize and efficiently train, especially in cases like SimulST, with further modifications required for stabilizing the policy head training and no guarantee of convergence (Gu et al. 2017).

Closely related to our work is the divergence-guided approach of DiG-SST (Chen et al. 2024). DiG-SST trains a lightweight policy module using the expected divergence between output distributions conditioned on partial versus complete input, estimated from a non-streaming S2TT model. This approach is efficient to train and directly optimizes for the SimulST task. Nevertheless, DiG-SST's formulation fails to make use of valuable information from ground truth labels when computing divergence scores. In REINA, we propose an improved formulation of a similar concept, yielding better streaming results.

# 3 Model

In this section, we introduce REINAStream, a low latency SimulST model trained on large-scale, open source data. We illustrate our architecture and loss functions in figure 1.

## 3.1 Policy Learning

To learn an effective READ/WRITE policy, we introduce a new loss function: Regularized Entropy INformation Adaptation (REINA). REINA enables us to adapt a non-streaming speech translation model into a streaming, SimulST model with minimal extra training.

First, we outline the problem more formally. Suppose that we are translating an input audio stream $a$ into a target language with a streaming chunk size of $j$ frames. Given a partial audio recording $a_t$ at frame $t$ and previously emitted tokens $s_1, s_2, \ldots, s_n$, we need to decide whether to produce token $s_{n+1}$ (WRITE) or wait for another audio chunk (READ). If we READ, we consume another frame of audio, giving us $a_{t+1}$, whereas if we WRITE, we gain a token, yielding the same audio $a_t$ but tokens $s_1, \ldots, s_n, s_{n+1}$. Our policy must learn to make READ/WRITE decisions that maximize translation quality while minimizing the latency with which we emit each token. This recursive setup appears to lend itself to a dynamic programming type of optimization, as in Seamless (Communication et al. 2023) or the

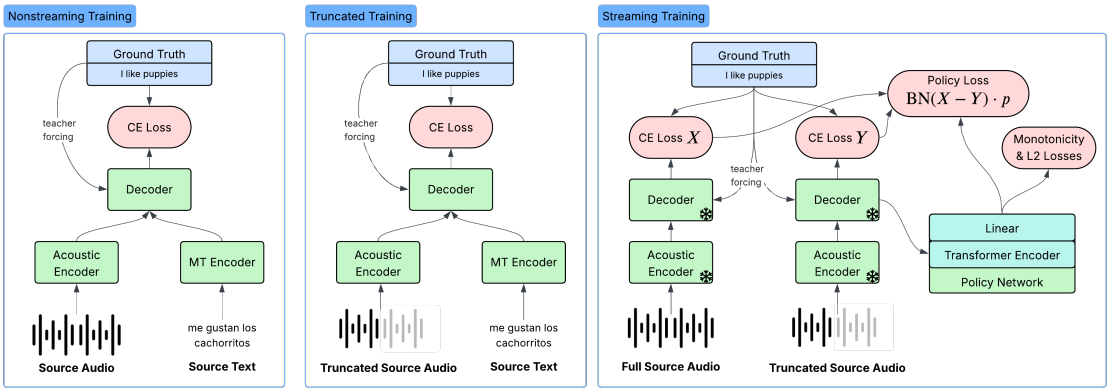



Figure 1: Non-streaming and streaming training procedures for REINAStream. For non-streaming training we use a trainable MT encoder to train on parallel NMT data. During streaming training we a) pass a full audio and truncated audio through the model, b) compute the cross-entropy (CE) loss of each, c) predict a policy using the policy network on top of the partial-audio output of the decoder, and finally d) calculate the REINA loss using the CE terms and policy predictions.

Transducer architecture (Graves 2012). However, in practice, optimizing over all possible READ/WRITE sequences results in expensive, numerically unstable training.

Instead, we start from a core idea: we should wait for more audio (i.e. READ) if and only if we gain information by doing so. We formalize this notion using mutual information theory. Given audio $a$ of length $T$ and ground truth translation token sequence $S = (s_1, \ldots, s_N)$, after writing $n < N$ tokens and listening to $t < T$ timesteps of audio, we can express the information gained about the next token $s_{n+1}$ by waiting for the rest of the input audio as

$$\mathcal{F}(a, S, n, t) := I(s_{n+1}; a_T, S_n) - I(s_{n+1}; a_t, S_n) \quad (1)$$

where $I$ is the symbol for mutual information. We can then construct an ideal READ/WRITE policy $\pi_\alpha$ on top of this quantity: $\pi_\alpha(a, S, n, t)$ returns READ when $\mathcal{F}(a, S, n, t) > \alpha$ and WRITE otherwise. We can then adjust $\alpha$ to control the latency quality tradeoff. This policy READs exactly when the information gained exceeds a given threshold. We can rewrite $\mathcal{F}(a, S, n, t)$ using mutual information equations (Barber and Agakov 2004) as follows

$$\begin{aligned}
\mathcal{F}(a, S, n, t) &:= I(s_{n+1}; a_T, S_n) - I(s_{n+1}; a_t, S_n) \\
&= H(s_{n+1}) - H(s_{n+1}|a_T, S_n) \\
&- [H(s_{n+1}) - H(s_{n+1}|a_t, S_n)] \\
&= H(s_{n+1}|a_t, S_n) - H(s_{n+1}|a_T, S_n) \\
&= \mathbb{E}\left[\log p(s_{n+1}|a_T, S_n) - \log p(s_{n+1}|a_t, S_n)\right] \quad (2)
\end{aligned}$$

Although we might not have access to $\log p(s_{n+1}|a_T, S_n)$ or $\log p(s_{n+1}|a_t, S_n)$, we are able to estimate these via random sampling of the log-probabilities of the base S2TT model as $\log \hat{p}(s_{n+1}|a_T, S_n)$ and $\log \hat{p}(s_{n+1}|a_t, S_n)$, which are the log-probabilities of the next label token computed when passing the full audio and partial audio through the model respectively. Note that these are also the negatives of the cross-entropy losses obtained when running the model on the full and partial audio.

We now have an estimate $\hat{\mathcal{F}}(a, S, n, t)$. However, this estimate cannot be computed during inference time as it requires access to the full target text, so we instead formulate a heuristic to estimate the information gain. We train a model parameterized by $\theta$ to estimate a heuristic $q_\theta = q(a, S, n, t|\theta)$ that yields policy $\hat{\pi}_\alpha$, which returns READ if and only if $q_\theta > \alpha$. We train $q_\theta$ to strongly correlate with $\mathcal{F}(a, S, n, t)$ by maximizing the covariance between $q_\theta$ and the estimate $\hat{\mathcal{F}}(a, S, n, t)$. This gives us the following optimization problem:

$$\begin{aligned}
&\max_\theta \Big(\mathrm{Cov}(q_\theta, \hat{\mathcal{F}}(a, S, n, t))\Big) \\
&= \max_\theta \Big(\mathbb{E}\left[q_\theta \cdot \hat{\mathcal{F}}(a, S, n, t)\right] - \\
&\qquad \mathbb{E}\left[q_\theta\right] \cdot \mathbb{E}\left[\hat{\mathcal{F}}(a, S, n, t)\right]\Big) \quad (3)
\end{aligned}$$

We can simplify this expression by normalizing the information gain estimate $\hat{\mathcal{F}}(a, S, n, t)$ to have zero mean, making $\mathbb{E}\left[q_\theta\right] \cdot \mathbb{E}\left[\hat{\mathcal{F}}(a, S, n, t)\right]$ evaluate to 0. We achieve this at train time by normalizing over each batch, resulting in the final optimization problem:

$$\begin{aligned}
&\max_\theta \Big[\mathbb{E}\left[q_\theta \cdot \mathrm{BN}\Big(\hat{\mathcal{F}}(a, S, n, t)\Big)\right]\Big] \\
&= \min_\theta \Big[q_\theta \cdot \mathrm{BN}\big[\log \hat{p_t}^{s_{n+1}} - \log \hat{p_T}^{s_{n+1}}\big]\Big] \quad (4)
\end{aligned}$$

where BN is the batchnorm operator, and we apply a shorthand $\log \hat{p_T}^{s_{n+1}} = \log \hat{p}(s_{n+1}|a_T, S_n)$ and $\log \hat{p_t}^{s_{n+1}} = \log \hat{p}(s_{n+1}|a_t, S_n)$ for brevity.

The loss that optimizes the above optimization problem is

$$\mathcal{L}_p = \frac{1}{N} \sum_{n=0}^{N-1} q_\theta^n \cdot \mathrm{BN}\big[\log \hat{p_t}^{s_{n+1}} - \log \hat{p_T}^{s_{n+1}})\big] \quad (5)$$

where $q_\theta^n = q_\theta(a, S, n, t|\theta)$. This loss maximizes the covariance between our estimate of information gain $\hat{\mathcal{F}}$ and our

heuristic estimator network $q_\theta$. This is the "Policy Loss" in figure 1. Since $\log \hat{p_t}(s_{n+1})$ and $\log \hat{p_T}(s_{n+1})$ are the negative cross-entropy losses from the decoder given partial and full audio respectively, in the diagram we write $\mathcal{L}_p$ as a difference of cross-entropy loss terms.

To ensure stable training and a reasonable learned policy, we add monotonicity and $L_2$ regularization terms. First, we note that at inference time, after predicting a READ, we predict no further tokens. Therefore, any WRITEs after a READ are semantically meaningless.

To better align train and inference, we add a weak monotonicity constraint on $q_\theta$ to encourage the probability of READ to increase uniformly across each token sequence $S$:

$$\mathcal{L}_m = \frac{1}{N}\sum_{n=1}^{N}\Big[\max\Big(\max_{m<n}\{q_\theta^m\} - q_\theta^n - \epsilon, 0\Big)\Big] \quad (6)$$

The loss function $\mathcal{L}_m$ serves as a regularization term to shape the learned heuristic $q_\theta$ used in the policy $\hat{\pi}_\alpha(a, S, n, t)$. It encourages the sequence of $q_\theta$ values for target text tokens, ordered by index $n$, to be approximately non-decreasing ($q_\theta^n \geq \max_{m<n}\{q_\theta^m\} - \epsilon$), an inductive bias distinct from merely estimating information gain. This imposed monotonicity biases the policy towards a "commitment" behavior, encouraging it to commit to stop predicting new tokens and READ new audio instead once the threshold is crossed for an earlier token.

Finally, we add a simple $L_2$ regularization penalty: $\mathcal{L}_r = \frac{1}{N}\sum_{n=1}^{N}(q_\theta^n)^2$. We find this is required to prevent $q_\theta$ values from exploding to infinity during training. Putting all the terms together, we get the full REINA loss: $\mathcal{L}_{\text{REINA}} = \mathcal{L}_p + \mathcal{L}_m + \lambda\mathcal{L}_r$. In our work, we set $\lambda = 0.05$, but find that final model performance is not very sensitive to changes in $\lambda$. We will cover how we train with $\mathcal{L}_{\text{REINA}}$ in section 3.4.

### 3.2 Non-Streaming Architecture

Next, we describe the architecture of our base non-streaming S2TT model, which is also outlined in figure 1. The architecture comprises of an acoustic encoder and a text decoder. At train-time, we also use an extra text encoder to facilitate a MT training task.

For the acoustic encoder, we adopt Whisper Medium (Radford et al. 2023) and do not freeze its weights during training. The randomly initialized transformer decoder performs cross-attention over the acoustic encoder's final layer hidden states and predicts text tokens. We adopt the generic multilingual tokenizer from Mistral 7B (Jiang et al. 2023) but learn our own embedding dictionary. We augment the vocabulary with language ids such as <en>or<fr>so we can direct the decoder to predict tokens of a specific target language by prefixing the token sequence with the language id. We apply a learned positional encoding similar to Time2Vec (Kazemi et al. 2019) on the acoustic encoder outputs to give the decoder a notion of sequence ordering. The decoder is trained with a cross-entropy loss.

At train-time, to support MT loss calculation, we add a randomly initialized, trainable T5 text encoder (Raffel et al. 2020). We pass source-language text through and then have the decoder cross-attend to the T5 last layer hidden states while predicting target language text. This is facilitates a machine translation task designed to improve the quality of the decoder by making use of paired MT data.

Whisper Medium contains 307M parameters, the text decoder has 101M, and the MT encoder has 38M for a total of 445M trainable parameters at train time and 408M at inference-time. While this is a larger parameter count than academic works like Dig-SST (Chen et al. 2024), StreamSpeech (Zhang et al. 2024), or Stream-Att (Papi et al. 2024), it is still much smaller than most industry systems like SeamlessM4T (Communication et al. 2023) or Hibiki (Labiausse et al. 2025). While higher parameter counts demonstrably increase translation quality, they also make it difficult to train and deploy models to large numbers of users in the wild. We target the middle ground between small and large systems in literature, yielding a model with both the translation quality and computational efficiency to be usable in real-world chat settings.

### 3.3 Streaming Architecture

We augment the non-streaming model with a policy network that makes binary READ/WRITE decisions for each output token. This is the architecture component that REINA serves to train. The policy network is a small transformer encoder applied to the last layer hidden states from the decoder. We apply a single linear layer on top of the transformer with output dimension 1 and sigmoid activation in order to make the binary READ/WRITE decisions. We apply a causal attention mask on the policy network at both train and inference time. The policy network contains only 6M trainable parameters, bringing the total parameter count for the streaming model at train and inference time to 414M.

### 3.4 Training

We train REINAStream in 3 stages: 1) Learn non-streaming S2TT 2) Adapt to truncated audios 3) Learn a streaming policy.

In the first stage, we train on several tasks at once in order to effectively leverage available data to train the speech translation model. All data samples contain some subset of the following information: source language $l_i^s$, source language audio $a_i$, source language transcription $T_i^s$, target language $l_i^t$, and target language transcription $T_i^t$. Using this data, we have three training tasks.

**ASR** For samples with $(a_i, l_i^s, T_i^s)$, we pass $a_i$ through the acoustic encoder, then decode to tokens in source language $l_i^s$. We compute cross-entropy loss using label transcript $T_i^s$ yielding loss $\mathcal{L}_{asr}$.

**NMT** For samples with $(T_i^s, l_i^t, T_i^t)$, we pass the source transcript through the T5 text encoder and then decode into target language $l_i^t$. We compute the cross-entropy loss using label transcript $T_i^t$ yielding loss $\mathcal{L}_{nmt}$.

**S2TT** For samples with $(a_i, l_i^t, T_i^t)$, we pass $a_i$ through the acoustic encoder, then decode into target language $l_i^t$. We compute cross-entropy loss using label transcript $T_i^t$ yielding loss $\mathcal{L}_{s2tt}$.

For the first stage training, we mix data supporting all tasks into every batch and minimize a sum of all the losses:

$\mathcal{L} = \mathcal{L}_{asr} + \mathcal{L}_{nmt} + \mathcal{L}_{s2tt}$.

Before training the policy network on top of the base model, we add a second step to ensure a higher quality estimation of $\log \hat{p}(s_{n+1}|S_n, a_t)$ with partial audios $a_t$ for the policy loss stage. In this phase, we fine-tune the speech translation model on randomly truncated audios using the same loss function $\mathcal{L}$.

Lastly, we train the policy network by minimizing $\mathcal{L}_{\text{REINA}}$ and freezing all other parameters. We only train on S2TT data samples as our goal is to learn a policy best for streaming speech translation rather than streaming ASR or MT.

### 3.5 Data

We seek to bridge the gap between models trained on large-scale, proprietary datasets and those trained on small-scale open-source data. We leverage a variety of publicly available data sources plus synthetic data generation to produce a large-scale training set.

In this iteration, we focus only on en→de, fr, es and de, fr, es→en language directions because of their data availability. In future work, we plan to expand to lower resourced languages. For audio datasets, we draw from Multilingual Librispeech (MLS) (Pratap et al. 2020), Mosel (Gaido et al. 2024), CVSS-C (Jia et al. 2022), MUST-C (Di Gangi et al. 2019). We list details on these datasets in table 1. We further augment the MLS dataset by translating its transcripts using an in-house NMT model to produce S2TT data to train on. We also augment our dataset with text-to-text MT training data from CCMatrix (Schwenk et al. 2021). We use 10M samples per language pair from CCMatrix for a total of 60M samples.

### 3.6 Inference Policy

We use streaming beam search to perform inference. We split input audios into $0.25$s chunks and inference the model on all audio up to the current chunk in sequence.

We first pick a policy threshold $\alpha$ to control the quality-latency tradeoff while streaming. Each iteration of the search, we run the policy network on all beams and kill those with predictions less than $\alpha$. If the total number of waited beams exceeds the beam size times a patience factor, or all beams are waited at once, we end the search. After ending the search, we return the killed beam with the highest average log probability.

Once we have reached the end of the input audio, we stop using the policy network and beam search until we hit the EOS token, using the same patience factor logic to decide when to end the search.

## 4 Experiments

In this section, we detail how we ran our experiments and present our results compared with existing work.

### 4.1 Experimental Setup

To train REINAStream, we follow the procedure outlined in Section 3.4, beginning with the non-streaming S2TT model. The model's text decoder is a 16-layer transformer with a model dimension of 512, 8 attention heads, a feedforward multiplier of 4, and uses label smoothing and dropout rates of 0.1. We train this initial model (Stage 1) for 5 days on 24 A100-80G GPUs using an AdamW optimizer (Loshchilov and Hutter 2019). The training configuration includes a fixed learning rate of $10^{-4}$, a weight decay of $10^{-4}$, and gradient clipping set to 10.0, with an effective batch size of 768. The data for this stage is a mixture of MUST-C (ratio 1), CVSS (ratio 1), MLS (en → X) (ratio 2), and MLS (X → en), CC-Matrix, and Mosel (all at ratio 4). For the second truncated stage (Stage 2), we use the exact same architecture and training configuration for 2 days, but train on a data mix of 20% full audios and 80% randomly truncated audios.

Lastly, we perform the REINA policy training (Stage 3). The REINA policy network is a smaller, 2-layer transformer with a 512 dimension embedding, 4 attention heads, and a feedforward multiplier of 4. It is regularized with a monotonicity loss with epsilon ($\epsilon$) of 0.5 and an L2 Regularization weight ($\lambda$) of 0.05. For this stage, we use an inverse square root learning rate scheduler with 5k warmup steps. We also adjust the dataset mixing ratios, increasing the ratios for MUST-C and CVSS to 2, and both directions of MLS to 6. This final training stage completes 20 epochs in under 12 hours.

We also re-implement DiG-SST's divergence-based loss from (Chen et al. 2024) based on the description in the paper and train the policy network with that loss using the same configuration as we train REINA.

We evaluate on language pairs {fr, de, es} → en on CVSS-C, and on language pairs en → {fr, de, es} on MUST-C. We use a beam size of 3 with no length penalty, a streaming chunk size of 0.25s, and a patience factor of 3. We sweep across several thresholds for the policy network in order to measure our model's tradeoff between latency and accuracy. Choosing the right thresholds to obtain an informative sweep is done purely through trial and error.

We compare to existing works on each dataset that we believe to be at or near to state-of-the-art at the time of writing. On MUST-C, we compare to Dig-SST (Chen et al. 2024), the work in literature closest to ours, and another strong SimulST competitor called DiSeg (Zhang and Feng 2023). On CVSS-C, we compare to StreamSpeech (Zhang et al. 2024) and SimulS2S-LLM (Deng et al. 2025a), a recent work showing strong simultaneous S2ST performance on CVSS with a small model. As StreamSpeech only reports ASR-BLEU whereas we report text BLEU, we are unable to make a fair comparison. That said, StreamSpeech is a stronger system than most in the literature on CVSS-C, and we outperform SimulS2S-LLM which outperforms StreamSpeech, so we show the results nontheless. For all comparisons, we use self-reported results from the original papers.

We also perform several ablations on the REINA training stage of REINAStream to demonstrate the utility of the different parts of the REINA loss and make fairer comparisons. We train five model variants:

**REINA** Our standard training procedure including all S2TT training datasets with the standard REINA loss function.

**REINA w/o monotonicity** Just like **REINA** but without the monotonicity term in the loss. Trained on all S2TT

| Split | Dataset | Source Language: en | | | | Source Language: de | | Source Language: es | | Source Language: fr | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Target: de | Target: es | Target: fr | Target: en | Target: en | Target: de | Target: en | Target: es | Target: en | Target: fr |
| Train | MLS | 13,789 | 13,785 | 13,787 | 41360 | 1,637 | 1,637 | 713 | 713 | 984 | 984 |
| | Must-C | 386 | 476 | 468 | 1330 | - | - | - | - | - | - |
| | CVSS-C | - | - | - | - | 184 | 184 | 113 | 113 | 264 | 264 |
| | MOSEL | - | - | - | 19,245 | - | 22,804 | - | 19,373 | - | 22,835 |
| Dev | Must-C | 2.47 | 2.49 | 2.49 | - | - | - | - | - | - | - |
| | CVSS-C | - | - | - | - | 21 | - | 22 | - | 22 | - |
| Test | Must-C | 4 | 4 | 4 | - | - | - | - | - | - | - |
| | CVSS-C | - | - | - | - | 22 | - | 23 | - | 23 | - |

Table 1: Consolidated Dataset Hours (Source Audio) by Split and Dataset, Grouped by Source and Target Language.

datasets.

**REINA (MUST-C only)** We train the policy network with the full REINA loss on only the MUST-C dataset for a fairer comparison to Dig-SST.

**REINA w/o truncated training** Just like **REINA** but skipping the truncated training stage. Trained on only the MUST-C dataset for comparison to Dig-SST.

**Dig-SST (Our impl. MUST-C only)** We train our own implementation of the DiG-SST on top of the non-streaming REINAStream model on only the MUST-C dataset.

## 4.2 Evaluation Metrics

We measure translation accuracy using BLEU as implemented in the SacreBLEU package (Post 2018), Average Lag (AL) and Length-Adaptive Average Lag (LAAL) (Papi et al. 2022), which we implement ourselves based on the original paper (Ma et al. 2020a). Most existing works plot AL vs BLEU curves by interpolating between several (AL, BLEU) points generated by evaluating their model with different streaming settings (Chen et al. 2024) (Papi et al. 2024) (Zhang et al. 2024). We contend that this evaluation is not sufficient adequately disentangle a model's non-streaming translation quality from its streaming ability. We observe that many comparisons in the literature pitch models with superior non-streaming BLEU as being better at streaming due to having a higher BLEU vs AL curve, when in reality the difference may be accounted for entirely by the non-streaming BLEU difference. Such comparisons do little to show the quality of the actual READ/WRITE policy.

We wish to show that REINA adapts S2TT models of any non-streaming performance to SimulST models with minimal degradation in translation quality, and compare against other methods that modify a non-streaming model to be streaming. To this end, we introduce Normalized Streaming Efficiency (NoSE), a metric to measure streaming performance across the entire quality/latency Pareto frontier normalized by non-streaming translation quality. NoSE is the measure of the area under the AL/BLEU curve, bounded on the left and right by $x < y$ respectively, divided by the area under the non-streaming BLEU line. It is important to note that while we require the $x$ and $y$ bounds to ensure the metric is well-defined, NoSE is heavily dependent on them. For our analysis, we pick the smallest $x$ and largest $y$ for which our work and the works we compare to all have reported values, yielding the widest possible range for which all models have a defined AL/BLEU curve. We recommend that future works using NoSE report the bounds used in their calculations.

## 4.3 Results

We present our NoSe scores in table 2 and selected operating points in table 3 for MUST-C and CVSS results respectively, with full AL-BLEU tradeoff curves in figure 2.

Unfortunately, CVSS-C is not commonly evaluated against in SimulST literature, so we are unable to compare to many other works. However, with an average utterance length of 4.9 seconds, CVSS is the only dataset out of the three comprising primarily shorter audios, which are quite common in conversational SimulST use-cases. This makes it an important benchmark for SimulST systems.

| **MUST-C** | | | |
|---|---|---|---|
| Model | en→de | en→fr | en→es |
| *Bounds* $[x, y]$ | [1.102, 1.965] | [1.187, 1.656] | [1.144, 1.416] |
| Dig-SST (Original) | 0.888 | 0.903 | 0.879 |
| DiSeg | 0.838 | - | 0.774 |
| Dig-SST (Our impl. MUST-C only) | 0.665 | 0.774 | 0.607 |
| EDAtt | 0.704 | - | 0.740 |
| REINA (MUST-C only) | **0.940** | **0.953** | **0.960** |
| REINA | 0.925 | 0.944 | 0.952 |
| REINA w/o monotonicity | 0.899 | 0.920 | 0.909 |
| **CVSS** | | | |
| Model | de→en | fr→en | es→en |
| *Bounds* $[x, y]$ | [1.955, 5.039] | [1.637, 5.169] | [1.806, 5.587] |
| StreamSpeech* | 0.842 | 0.886 | 0.837 |
| REINA w/o monotonicity | **0.976** | 0.980 | **0.982** |
| REINA | 0.974 | **0.983** | 0.981 |

Table 2: NoSE (↑) values for the MUST-C and CVSS-C datasets. Bounds $[x, y]$ for NoSE are specified per language pair. Best values per language pair are bolded. Results for other works are taken from their corresponding papers. *Note StreamSpeech only reports ASR-BLEU, so values for StreamSpeech represent ASR-BLEU scores rather than BLEU. SimulS2S-LLM is not included here as they do not publish their offline BLEU performance.

**Quantitative results** On both MUST-C and CVSS-C, REINA outperforms all competing methods on all language splits at low latencies. Significantly, this holds for the "MUST-C only" model with policy network trained only on MUST-C, as well as the REINA model trained on all datasets, demonstrating our streaming performance gains are

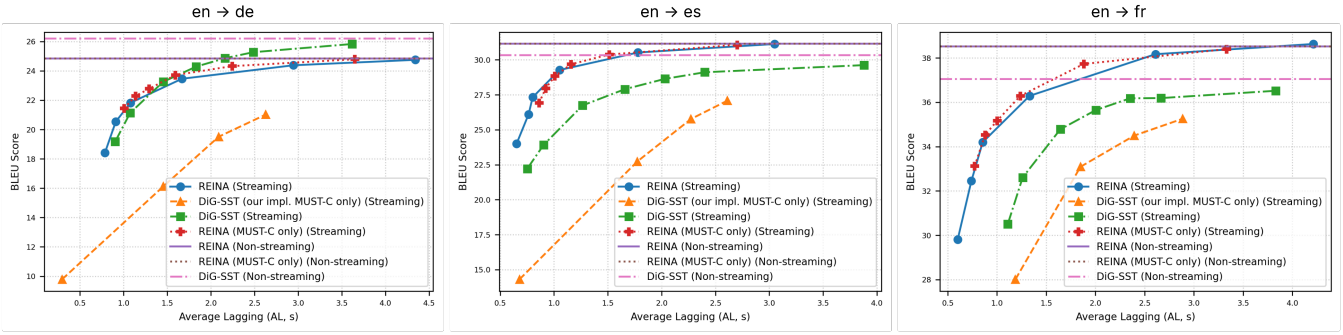



Figure 2: Average Lagging (AL) vs. BLEU score on MUST-C. Horizontal lines represent non-streaming performance.

not merely attributable to increased data scale when training the policy, but come from the improved objective. The only exception to this, as seen in table 3, is German, where DigSST is slightly better at higher latencies than REINA. Still, REINA excels at lower latency streaming, even when its non-streaming BLEU is lower than competitors.

The MUST-C-only REINA model yields NoSE scores 3.0% higher than Dig-SST and 8.9% higher than DiSeg, highlighting REINA's mutual information formulation advantage. Our implementation of DiG-SST performs far below every other model in evals, suggesting we missed details during reproduction of results.

**Ablations on Monotonicity Loss** In figure 3, we observe that the REINA model outperforms the REINA w/o monotonicity model exclusively at low latencies, indicating that monotonicity is useful on the most aggressive streaming settings. For example, at about 35 BLEU, AL decreases from 1.95 to 1.57, a 19% improvement. We hypothesize this is because monotonicity forces the policy to decide on a clear boundary of when to READ when the information gain waffles between timesteps. Ultimately, we find the monotonicity loss improves low-latency streaming with a negligible impact to overall streaming efficiency.

| | German (En→De) | | | Spanish (En→Es) | | | French (En→Fr) | | |
|---|---|---|---|---|---|---|---|---|---|
| Model | AL ↓ | LAAL ↓ | BLEU ↑ | AL ↓ | LAAL ↓ | BLEU ↑ | AL ↓ | LAAL ↓ | BLEU ↑ |
| REINA | 1.01 | 1.10 | 21.44 | 0.86 | 0.94 | 26.92 | 0.77 | 0.87 | 33.13 |
| DiG-SST | 1.08 | – | 21.13 | 0.90 | – | 23.92 | 1.11 | – | 30.51 |
| EDAtt | 1.04 | 1.20 | 19.10 | 0.95 | 1.24 | 23.00 | – | – | – |
| REINA | 1.59 | 1.67 | 23.71 | 1.16 | 1.24 | 29.68 | 1.24 | 1.32 | 36.29 |
| DiG-SST | 1.45 | – | 23.25 | 1.27 | – | 26.74 | 1.26 | – | 32.61 |
| EDAtt | 1.34 | 1.46 | 21.60 | 1.28 | 1.52 | 26.60 | – | – | – |
| REINA | 2.24 | 2.30 | 24.32 | 1.51 | 1.57 | 30.38 | 1.88 | 1.93 | 37.73 |
| DiG-SST | 1.83 | – | 24.29 | 1.66 | – | 27.90 | 2.00 | – | 35.65 |
| EDAtt | 2.26 | 2.33 | 25.60 | 1.52 | 1.74 | 27.80 | – | – | – |
| REINA (m) | 3.65 | 3.67 | 24.79 | 2.70 | 2.73 | 31.07 | 3.33 | 3.35 | 38.40 |
| DIGSST (r) | 3.62 | – | 25.83 | 2.40 | – | 29.11 | 3.83 | – | 36.52 |
| EDAtt (m) | 2.74 | 2.80 | 26.30 | 2.14 | 2.34 | 29.20 | – | – | – |

Table 3: Comparison of streaming translation models for MUST-C on En→{De, Es, Fr} on various operating points. DiG-SST does not report LAAL values.

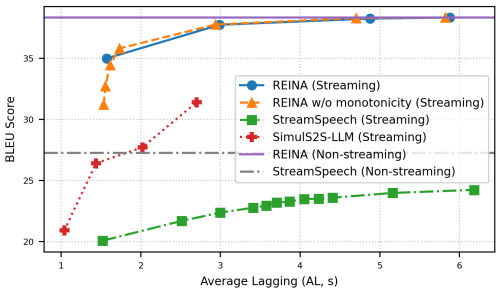



Figure 3: AL/BLEU curve on Es→En split of the CVSS-C dataset. We report ASR-BLEU only for StreamSpeech.

| | MUST-C | | |
|---|---|---|---|
| Model | en→de | en→fr | en→es |
| *Bounds* $[x, y]$ | [1.219, 1.606] | [1.068, 1.468] | [1.276, 1.686] |
| REINA | 0.932 | 0.942 | 0.971 |
| REINA (No Stage 2 Trunc.) | 0.840 | 0.839 | 0.895 |

Table 4: NoSE (↑) values ablating non-truncated training on MUST-C .

**Ablations on Truncated Training** Table 4 shows the importance of stage 2 training, suggesting that mutual information formulation of REINA requires a good estimate of $\log \hat{p}(s_{n+1}|a_t, S_n)$, as skipping that stage leads to a performance degradation in the trained policy.

## 5 Conclusion and Future Work

In this paper, we present a new method for SimulST. We introduce the REINA loss function that enables efficient conversion of non-streaming speech translation models into streaming ones. We conduct extensive experiments over several datasets, showing REINA outperforms the state of the art in SimulST conversion. We also propose a new metric, NoSE, to improve the state of evaluation of SimulST systems. Ultimately, we train a large-scale system entirely on open source or synthetically generated data, encouraging further research into scaling SimulST.

The next step to enable real-time crosslingual interaction is to extend REINAStream into a simultaneous speech to speech translation (SimulS2ST) model. This is achiev-

able by using a high quality, low latency, streaming text-to-speech model as a synthesizer. We are presently working on extending REINAStream to the SimulS2ST use-case.

## References


Arivazhagan, N.; Cherry, C.; Macherey, W.; Chiu, C.-C.; Yavuz, S.; Pang, R.; Li, W.; and Raffel, C. 2019. Monotonic Infinite Lookback Attention for Simultaneous Machine Translation. In Korhonen, A.; Traum, D.; and Màrquez, L., eds., *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 1313–1323. Florence, Italy: Association for Computational Linguistics.

Barber, D.; and Agakov, F. 2004. The im algorithm: a variational approach to information maximization. *Advances in neural information processing systems*, 16(320): 201.

Chen, X.; Fan, K.; Luo, W.; Zhang, L.; Zhao, L.; Liu, X.; and Huang, Z. 2024. Divergence-Guided Simultaneous Speech Translation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(16): 17799–17807.

Communication, S.; Barrault, L.; Chung, Y.-A.; Meglioli, M. C.; Dale, D.; Dong, N.; Duppenthaler, M.; Duquenne, P.-A.; Ellis, B.; Elsahar, H.; Haaheim, J.; Hoffman, J.; Hwang, M.-J.; Inaguma, H.; Klaiber, C.; Kulikov, I.; Li, P.; Licht, D.; Maillard, J.; Mavlyutov, R.; Rakotoarison, A.; Sadagopan, K. R.; Ramakrishnan, A.; Tran, T.; Wenzek, G.; Yang, Y.; Ye, E.; Evtimov, I.; Fernandez, P.; Gao, C.; Hansanti, P.; Kalbassi, E.; Kallet, A.; Kozhevnikov, A.; Gonzalez, G. M.; Roman, R. S.; Touret, C.; Wong, C.; Wood, C.; Yu, B.; Andrews, P.; Balioglu, C.; Chen, P.-J.; Costa-jussà, M. R.; Elbayad, M.; Gong, H.; Guzmán, F.; Heffernan, K.; Jain, S.; Kao, J.; Lee, A.; Ma, X.; Mourachko, A.; Peloquin, B.; Pino, J.; Popuri, S.; Ropers, C.; Saleem, S.; Schwenk, H.; Sun, A.; Tomasello, P.; Wang, C.; Wang, J.; Wang, S.; and Williamson, M. 2023. Seamless: Multilingual Expressive and Streaming Speech Translation. arXiv:2312.05187.

Deng, K.; Chen, W.; Chen, X.; and Woodland, P. C. 2025a. SimulS2S-LLM: Unlocking Simultaneous Inference of Speech LLMs for Speech-to-Speech Translation. *arXiv preprint arXiv:2504.15509*.

Deng, K.; Chen, W.; Chen, X.; and Woodland, P. C. 2025b. SimulS2S-LLM: Unlocking Simultaneous Inference of Speech LLMs for Speech-to-Speech Translation. arXiv:2504.15509.

Di Gangi, M. A.; Cattoni, R.; Bentivogli, L.; Negri, M.; and Turchi, M. 2019. MuST-C: a Multilingual Speech Translation Corpus. In Burstein, J.; Doran, C.; and Solorio, T., eds., *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2012–2017. Minneapolis, Minnesota: Association for Computational Linguistics.

Fu, B.; Yu, D.; Liao, M.; Li, C.; Chen, Y.; Fan, K.; and Shi, X. 2025. Efficient and Adaptive Simultaneous Speech Translation with Fully Unidirectional Architecture. *arXiv preprint arXiv:2504.11809*.

Gaido, M.; Papi, S.; Bentivogli, L.; Brutti, A.; Cettolo, M.; Gretter, R.; Matassoni, M.; Nabih, M.; and Negri, M. 2024. MOSEL: 950,000 Hours of Speech Data for Open-Source Speech Foundation Model Training on EU Languages. In Al-Onaizan, Y.; Bansal, M.; and Chen, Y.-N., eds., *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 13934–13947. Miami, Florida, USA: Association for Computational Linguistics.

Graves, A. 2012. Sequence Transduction with Recurrent Neural Networks. arXiv:1211.3711.

Gu, J.; Neubig, G.; Cho, K.; and Li, V. O. 2017. Learning to Translate in Real-time with Neural Machine Translation. In Lapata, M.; Blunsom, P.; and Koller, A., eds., *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, 1053–1062. Valencia, Spain: Association for Computational Linguistics.

Jia, Y.; Ramanovich, M. T.; Wang, Q.; and Zen, H. 2022. CVSS Corpus and Massively Multilingual Speech-to-Speech Translation. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, 6691–6703.

Jiang, A. Q.; Sablayrolles, A.; Mensch, A.; Bamford, C.; Chaplot, D. S.; de las Casas, D.; Bressand, F.; Lengyel, G.; Lample, G.; Saulnier, L.; Lavaud, L. R.; Lachaux, M.-A.; Stock, P.; Scao, T. L.; Lavril, T.; Wang, T.; Lacroix, T.; and Sayed, W. E. 2023. Mistral 7B. arXiv:2310.06825.

Kazemi, S. M.; Goel, R.; Eghbali, S.; Ramanan, J.; Sahota, J.; Thakur, S.; Wu, S.; Smyth, C.; Poupart, P.; and Brubaker, M. A. 2019. Time2Vec: Learning a Vector Representation of Time. *CoRR*, abs/1907.05321.

Labiausse, T.; Mazaré, L.; Grave, E.; Pérez, P.; Défossez, A.; and Zeghidour, N. 2025. High-Fidelity Simultaneous Speech-To-Speech Translation. *arXiv preprint arXiv:2502.03382*.

Loshchilov, I.; and Hutter, F. 2019. Decoupled Weight Decay Regularization. In *International Conference on Learning Representations*.

Ma, X.; Dousti, M. J.; Wang, C.; Gu, J.; and Pino, J. 2020a. SIMULEVAL: An Evaluation Toolkit for Simultaneous Translation. In Liu, Q.; and Schlangen, D., eds., *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 144–150. Online: Association for Computational Linguistics.

Ma, X.; Pino, J.; and Koehn, P. 2020. SimulMT to SimulST: Adapting Simultaneous Text Translation to End-to-End Simultaneous Speech Translation. In Wong, K.-F.; Knight, K.; and Wu, H., eds., *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, 582–587. Suzhou, China: Association for Computational Linguistics.

Ma, X.; Pino, J. M.; Cross, J.; Puzon, L.; and Gu, J. 2020b. Monotonic Multihead Attention. In *International Conference on Learning Representations*.

Papi, S.; Gaido, M.; Negri, M.; and Bentivogli, L. 2024. StreamAtt: Direct Streaming Speech-to-Text Translation

with Attention-based Audio History Selection. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 3692–3707.

Papi, S.; Gaido, M.; Negri, M.; and Turchi, M. 2022. Over-Generation Cannot Be Rewarded: Length-Adaptive Average Lagging for Simultaneous Speech Translation. In Ive, J.; and Zhang, R., eds., *Proceedings of the Third Workshop on Automatic Simultaneous Translation*, 12–17. Online: Association for Computational Linguistics.

Papi, S.; Negri, M.; and Turchi, M. 2023. Attention as a Guide for Simultaneous Speech Translation. In Rogers, A.; Boyd-Graber, J.; and Okazaki, N., eds., *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 13340–13356. Toronto, Canada: Association for Computational Linguistics.

Peng, Y.; Tian, J.; Chen, W.; Arora, S.; Yan, B.; Sudo, Y.; Shakeel, M.; Choi, K.; Shi, J.; Chang, X.; et al. 2024. OWSM v3. 1: Better and Faster Open Whisper-Style Speech Models based on E-Branchformer. In *Proc. Interspeech 2024*, 352–356.

Post, M. 2018. A Call for Clarity in Reporting BLEU Scores. In Bojar, O.; Chatterjee, R.; Federmann, C.; Fishel, M.; Graham, Y.; Haddow, B.; Huck, M.; Yepes, A. J.; Koehn, P.; Monz, C.; Negri, M.; Névéol, A.; Neves, M.; Post, M.; Specia, L.; Turchi, M.; and Verspoor, K., eds., *Proceedings of the Third Conference on Machine Translation: Research Papers*, 186–191. Brussels, Belgium: Association for Computational Linguistics.

Pratap, V.; Xu, Q.; Sriram, A.; Synnaeve, G.; and Collobert, R. 2020. MLS: A Large-Scale Multilingual Dataset for Speech Research. In *Proc. Interspeech 2020*, 2757–2761.

Puvvada, K. C.; Żelasko, P.; Huang, H.; Hrinchuk, O.; Koluguri, N. R.; Dhawan, K.; Majumdar, S.; Rastorgueva, E.; Chen, Z.; Lavrukhin, V.; et al. 2024. Less is More: Accurate Speech Recognition & Translation without Web-Scale Data. In *Proc. Interspeech 2024*, 3964–3968.

Radford, A.; Kim, J. W.; Xu, T.; Brockman, G.; McLeavey, C.; and Sutskever, I. 2023. Robust speech recognition via large-scale weak supervision. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org.

Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; and Liu, P. J. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140): 1–67.

Schwenk, H.; Wenzek, G.; Edunov, S.; Grave, E.; Joulin, A.; and Fan, A. 2021. CCMatrix: Mining Billions of High-Quality Parallel Sentences on the Web. In Zong, C.; Xia, F.; Li, W.; and Navigli, R., eds., *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 6490–6500. Online: Association for Computational Linguistics.

Xue, J.; Wang, P.; Li, J.; Post, M.; and Gaur, Y. 2022. Large-Scale Streaming End-to-End Speech Translation with Neural Transducers. In *Proc. Interspeech 2022*, 3263–3267.

Ye, R.; Wang, M.; and Li, L. 2022. Cross-modal Contrastive Learning for Speech Translation. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 5099–5113.

Zhang, S.; Fang, Q.; Guo, S.; Ma, Z.; Zhang, M.; and Feng, Y. 2024. StreamSpeech: Simultaneous Speech-to-Speech Translation with Multi-task Learning. In *Proceedings of the 62th Annual Meeting of the Association for Computational Linguistics (Long Papers)*. Association for Computational Linguistics.

Zhang, S.; and Feng, Y. 2023. End-to-End Simultaneous Speech Translation with Differentiable Segmentation. In *Findings of the Association for Computational Linguistics: ACL 2023*, 7659–7680. Association for Computational Linguistics.

# Appendix

## A NoSE Score

To clarify our definition of the NoSE score, we provide a diagram in figure A.1.

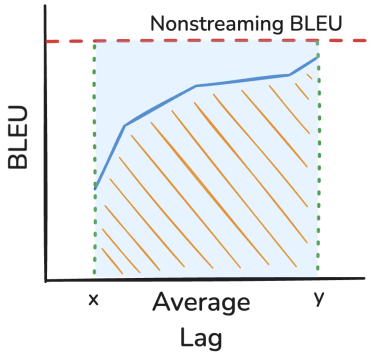



Figure A.1: We define NoSE as the area of the orange shaded region divided by the area of the blue rectangle.

## B Latency vs BLEU Evaluations

When evaluating our models on each dataset, we sweep across several policy thresholds. We find that the unique features of each dataset necessitate different thresholds to get the best results. In table B.1 we show the list of thresholds we used for inferencing each model on each dataset. Unfortunately, the only we have to determine these thresholds is trial and error. We had to perform several offline inference sweeps across thresholds before finding a sweep that a) allowed us to compare to other works b) contained points in the AL/BLEU curve that were representative of how our model might be likely to be used in production.

We show AL vs BLEU graphs for all evaluations that appear in the paper in figures B.1 and B.2.

## C Hyperparameters

We report model and dataset hyperparameters in table C.1.

## D EMMA Discussion

In our paper, we touch on the complexities of implementing streaming via monotonic attention as in the EMMA method used in (Communication et al. 2023). In this section, we wish to provide some more background on the computational and numerical challenges that arise from training with EMMA.

In our research, we implemented EMMA from scratch on top of a non-streaming REINAStream model. The train-time computations for EMMA requires computing a matrix of size [batch_size $\times$ attention_heads $\times$ num_text_tokens $\times$ audio_sequence_length $\times$ audio_sequence_length] within each cross-attention layer of the decoder. Seeing as the audio sequence length coming out of the whisper encoder is 1500, we use 8 attention heads, and a short token sequence may contain around 25 tokens, using fp32 precision, we require 2GB of VRAM for a single cross-attention layer at batch size 1. Ultimately, we were only able to train EMMA with batch size 1 and truncating the encoder output sequence length to 500, despite using A100-80G GPUs. This made for very slow, expensive training.

Furthermore, the EMMA estimation requires computing a cumulative product across the audio sequence length dimension. A cumulative product of 500 small floating point values is numerically unstable and often results in rounding to 0. Perhaps the original authors used a sum of log values instead.

Lastly, as EMMA computes a separate policy for every attention head of every cross-attention layer, it is unclear which one to use for the final inference policy. In the public inference code associated with (Communication et al. 2023), the layer to use for the policy is simply taken as an argument. Attention heads from that layer are aggregated via a max, min, or mean. Empirically, we found that some layers and heads learned useful policies, while the majority did not.

Due to these challenges implementing and training EMMA, we did not include it in our evaluations and pursued simpler streaming methods, motivating us to invent REINA.

We also note that many of the same issues arise with training neural transducer (Graves 2012) models. We also spent considerable effort implementing transducers for SimulST. As with EMMA, we found transducer methods to be excessively expensive to train and very hard to make converge.

## E FLEURS Evaluation

In this section, we present an attempt at comparing to Seamless on FLEURS. We construct the FLEURS dataset for en $\rightarrow$ de, en, fr and de, en, fr $\rightarrow$ en, deduplicating on unique source audio and target language. We trim the dataset using the same Silero VAD Seamless uses. When inferencing, we soon noticed that as our model is trained entirely on audios not trimmed with VAD, it tends to expect an ending silence and over-generates, resulting in decreased BLEU scores. We also attempted inferencing Seamless on our copy of FLEURS using their open source code, but ran into implementation difficulties due to scarce documentation.

In a final attempt at a comparison, we use Seamless' reported scores from their paper. We use VAD to trim our audios and then augment them with 2 seconds of white noise at the end to help our model. We tested this trick without VAD on our other datasets and found it has a negative impact on BLEU, but we found it was still an improvement over inferencing on VAD-trimmed audios on FLEURS.

We show NoSE scores in table E.1, graphs for the X $\rightarrow$ en direction in figure E.1, and graphs for the en $\rightarrow$ X direction in figure E.2. As our model is much smaller than Seamless, our non-streaming BLEU is unsurprisingly much lower. Of more interest are the NoSE scores, which show we are com-

Table B.1: Thresholds swept for generating AL vs. BLEU points for different models and datasets.

| Dataset | Model Name | Thresholds Swept |
|---|---|---|
| **MUST-C** | Dig-SST (Original) | Self-reported results |
| | DiSeg | Self-reported results |
| | EdAtt | Self-reported results |
| | Dig-SST (Our impl. MUST-C only) | [.475, .5, .51, .52, .53, .54] |
| | REINA (MUST-C only) | [.935, .94, .9425, .945, .9475, .95] |
| | REINA (No Truncation) (MUST-C only) | [.975, .976, .977, .978, .979, .980] |
| | REINA | [.97, .975, .976, .977, .978, .979] |
| **CVSS** | REINA | [.97, .975, .976, .977, .978, .979, .98, .983, .985, .987] |
| | REINA w/o monotonicity | [.976, .977, .978, .979, .98, .983, .985, .987] |
| | StreamSpeech | Self-reported results |
| | SimulS2S-LLM | Self-reported results |

Table C.1: Hyperparameters and Dataset Mixing Ratios used for training the S2TT model and the REINA policy network.

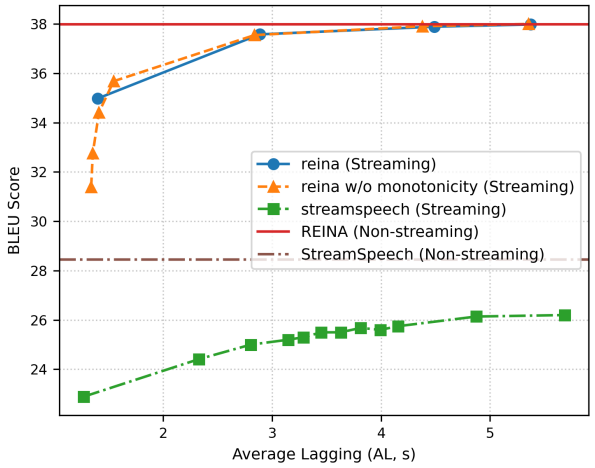
| Parameter / Setting | Value |
|---|---|
| **Text Decoder Configuration** | |
| Dimension | 512 |
| Attention Heads | 8 |
| Number of Layers | 16 |
| Feedforward Multiplier | 4 |
| Label Smoothing | 0.1 |
| Dropout | 0.1 |
| **Policy Network Configuration (REINA)** | |
| Monotonicity Epsilon ($\epsilon$)* | 0.5 |
| L2 Regularization Weight ($\lambda$) | 0.05 |
| Number of Layers | 2 |
| Attention Heads | 4 |
| Feedforward Multiplier | 4 |
| Dimension | 512 |
| **Training Configuration** | |
| Learning Rate | 0.0001 (fixed) |
| Weight Decay | 0.0001 |
| Gradient Clipping | 10.0 |
| Batch Size per Device | 16 |
| Gradient Accumulation Steps | 2 |
| Effective Batch Size | 768 |
| **Dataset Mixing: Base S2TT Model Training (Stage 1 & 2)** | |
| Dataset Name | Mixing Ratio |
| MUST-C (train split) | 1 |
| CVSS (train split) | 1 |
| MLS (en $\rightarrow$ X)** | 2 |
| MLS (X $\rightarrow$ en)** | 4 |
| CCMatrix | 4 |
| Mosel | 4 |
| **Dataset Mixing: REINA Policy Training (Stage 3)** | |
| Dataset Name | Mixing Ratio |
| MUST-C (train split) | 2 |
| CVSS (train split) | 2 |
| MLS (en $\rightarrow$ X)** | 6 |
| MLS (X $\rightarrow$ en)** | 6 |

* Corresponds to $\epsilon$ in the monotonicity loss (Equation 3).
** Derived from Multilingual Librispeech (MLS) dataset (see Section 3.5).

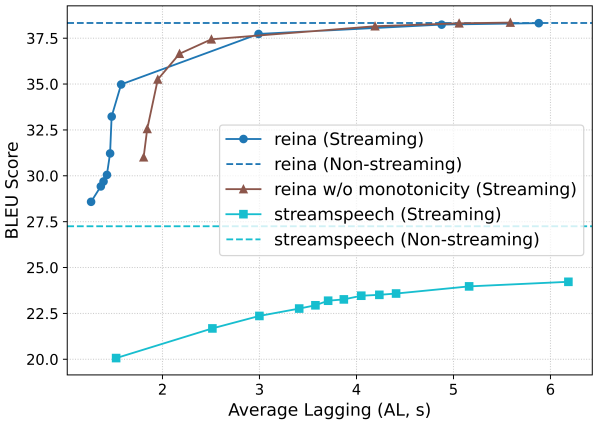
parable to Seamless on the X $\rightarrow$ en directions but worse on en $\rightarrow$ X. We believe this difference is primarily accounted for by the aforementioned VAD issues. We also note that REINA is vastly cheaper and easier to train than Seamless' EMMA streaming method (see appendix D), meaning that even with comparable streaming quality, REINA is in most cases preferable to EMMA.

| Dataset | Item / Model | en→de | en→fr | en→es | de→en | fr→en | es→en |
|---|---|---|---|---|---|---|---|
| | *Bounds* $[x, y]$ | [1.87, 2.05] | [1.71, 1.86] | [1.83, 1.99] | [1.68, 1.85] | [1.42, 1.55] | [1.36, 1.52] |
| **FLEURS** | Seamless | **0.914** | **0.951** | **0.960** | .924 | **.940** | **.936** |
| | REINAStream | 0.896 | 0.861 | 0.916 | **.943** | .866 | **.936** |

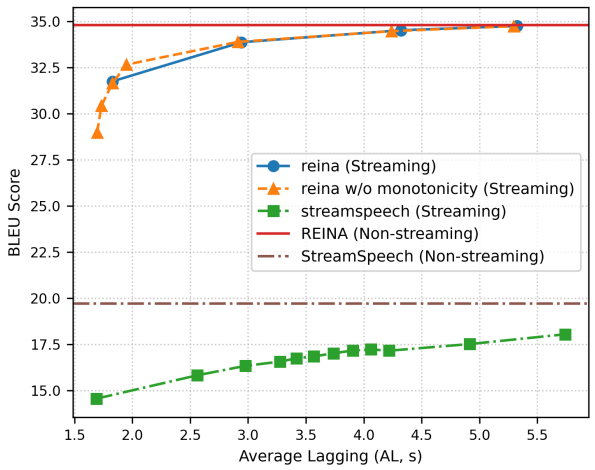
Table E.1: Comparison of NoSE (↑) values on FLEURS for Seamless and REINAStream.




(a) French → English (fr-en)

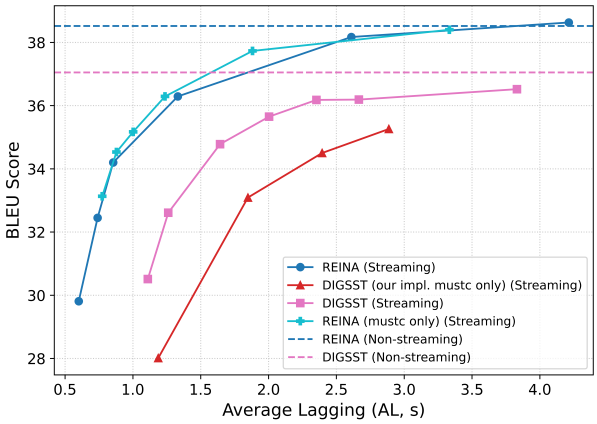



(b) Spanish → English (es-en)

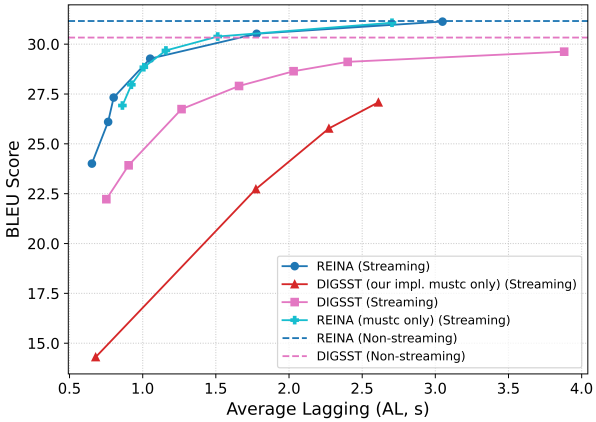



(c) German → English (de-en)

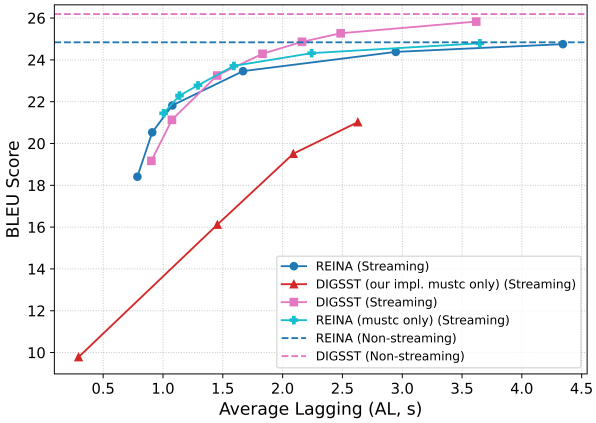
Figure B.1: Average Lagging (AL) vs. BLEU score on CVSS-C. Dotted lines represent non-streaming BLEU scores. Note that StreamSpeech only reports ASR-BLEU in their paper, so we report StreamSpeech's ASR-BLEU rather than BLEU.




(a) English → French (en-fr)

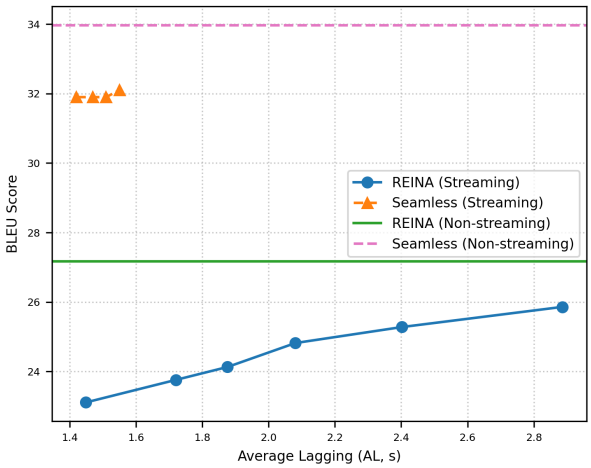



(b) English → Spanish (en-es)

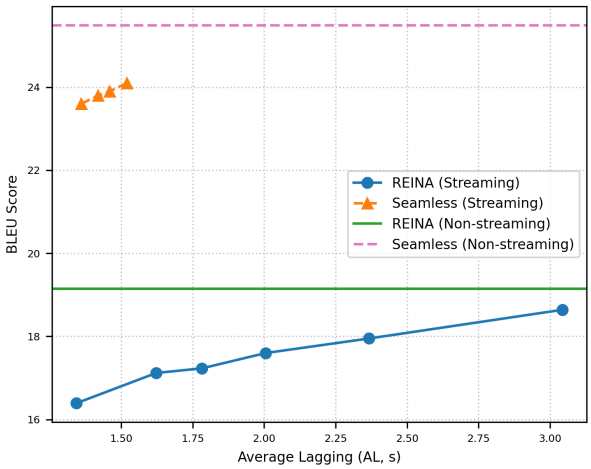



(c) English → German (en-de)

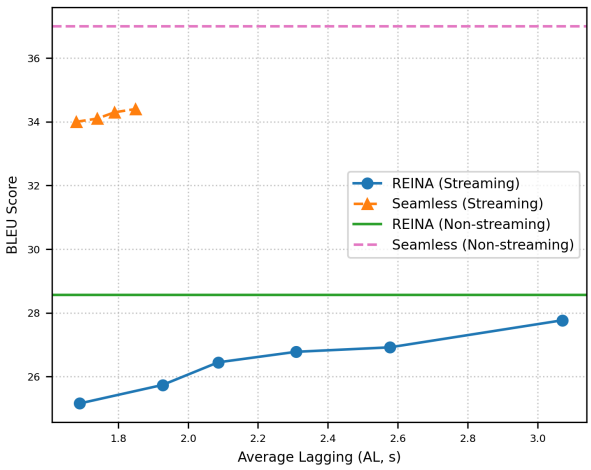
Figure B.2: Average Lagging (AL) vs. BLEU score on MUST-C. Dotted lines represent non-streaming BLEU scores.




(a) French → English (fr-en)

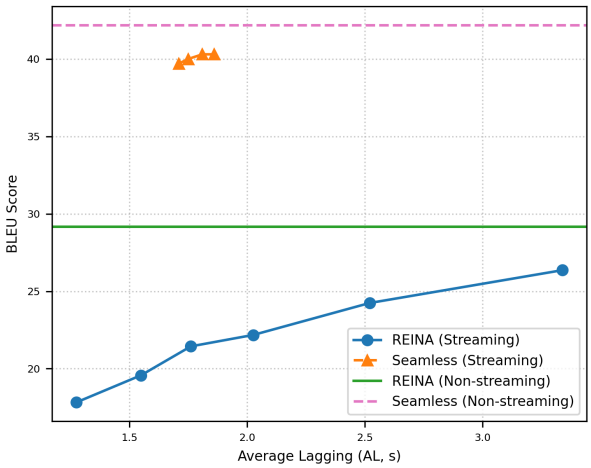



(b) Spanish → English (es-en)

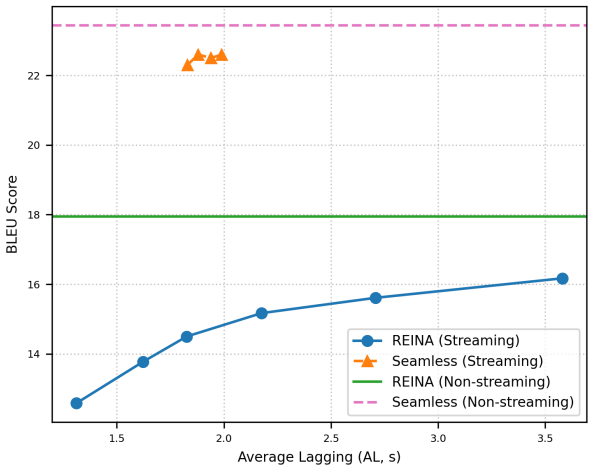



(c) German → English (de-en)

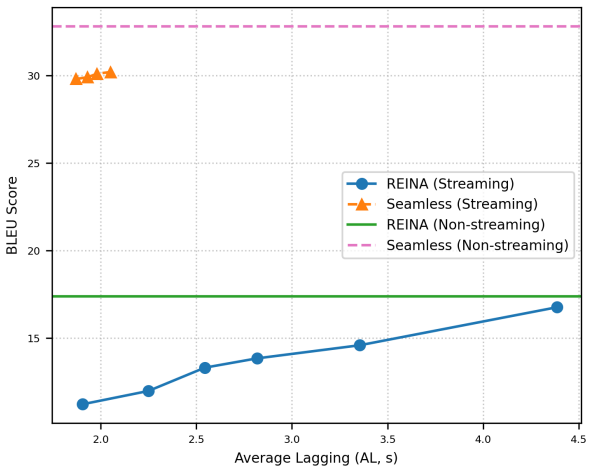
Figure E.1: Average Lagging (AL) vs. BLEU score on FLEURS X → en. Dotted lines represent non-streaming BLEU scores.




(a) English → French (en-fr)




(b) English → Spanish (en-es)




(c) English → German (en-de)

Figure E.2: Average Lagging (AL) vs. BLEU score on FLEURS en → X. Dotted lines represent non-streaming BLEU scores.